Supervised Hate Speech Detection of Social Media Text

Kevin Liu

1 Dataset Identification

For this study, I identified a dataset containing text messages and corresponding binary labels 1 or 0 which describe whether or not the text was hate speech or not [1][2].

The dataset itself had already been preprocessed using a few techniques. Preprocessing included removing punctuation, extra whitespace, links, user mentions, date and time values, accented numbers and characters, fixing grammatical errors and misspelled profanities. Additionally, the dataset had contractions expanded and numbers converted to word form to ensure that content was only of text form.

The preprocessed dataset I started with, "HateSpeechDataset.csv", contained 440,906 messages with 361594 messages labeled as non-hate, 79305 messages labeled as hate, and 7 additional invalid entries where the label was the string "Label" and the content of the message was the string "content". These 7 invalid entries were removed so that only valid data was used for training and testing. Figure 1 showcases the distribution of message lengths in the dataset after the removal of invalid samples. It can be seen that most messages are relatively short with some outlier messages being extremely long. In

Statistic	Value
Count	440899
Mean	226.119513
Standard Deviation	277.814219
Minimum	1
25%	62
50%	119
75%	269
Maximum	9998

Table 1: Descriptive statistics of valid data in dataset

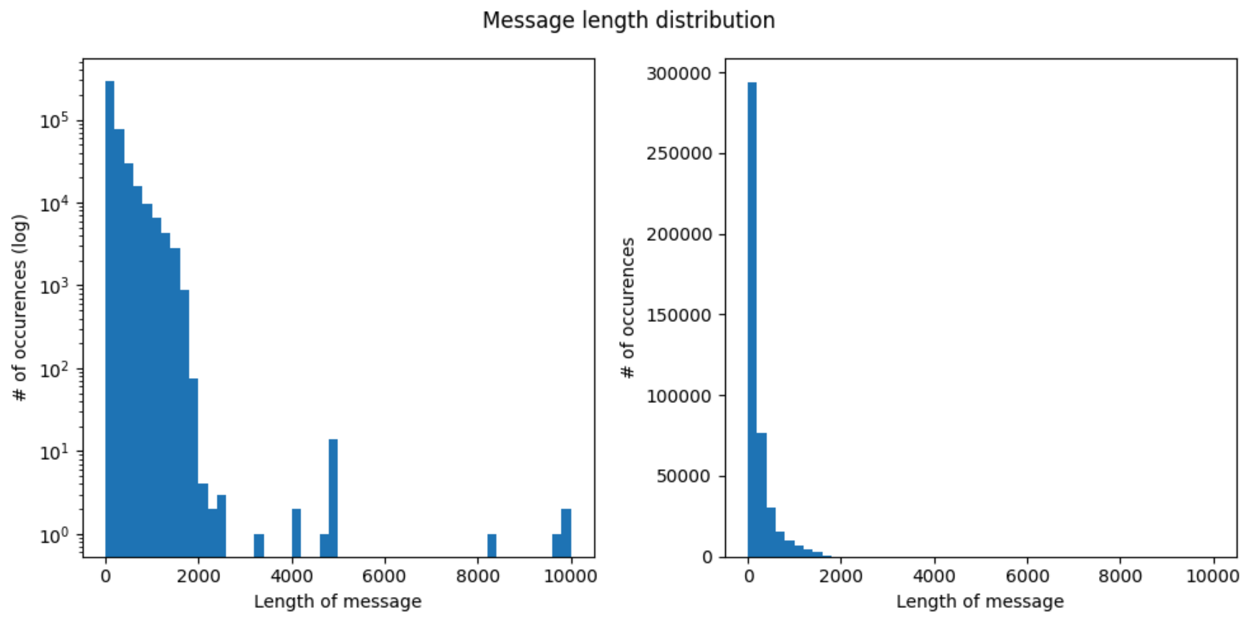

Figure 1: Distribution of message length in the dataset

Table 1, it can be seen that most messages are only a few hundred characters long, thus I removed samples that had messages longer than or equal to 300 characters long or were only 1 character long.

Some examples of outlier messages include "w" (length of 1) and "do not change it zzzzzz…" (length of 9998). This additional preprocessing reduced the size of the dataset from its original 440,906 samples to 343,379 samples with 271,100 positive labels and 72,279 negative labels.

2 Predictive Task Identification

Using this dataset, I decided to try to classify whether or not the message is hate speech or not. This is inline with the intended purpose of the dataset as the existing preprocessing gave each sample a label of 1 if the message was hate speech or a label of 0 if the message was not hate speech.

I'll use accuracy as the main evaluation metric for each model, but other metrics like precision, recall, F1 score, and balanced accuracy will also be included in the final results comparison. I'll use a 50-25-25 train, validation, test split on the original dataset; model training will be done using the training set, hyperparameter optimization by comparing model performance on the validation set, and finally, the validity of the models' predictions will be ensured by comparing final performance on the test set. A baseline performance that I can compare model performance to the accuracy of a random classifier which would have an average accuracy of 0.5 with 2 different label classes.

For features, the three representations I'll be constructing are two bag of words feature representations, one using the N most common words and one using the N highest TF-IDF words, and one dimensionality-reduction based feature representation using gensim's word2vec to create word embeddings.

The bag of words feature using the N most popular words is created by iterating over all messages in the training dataset, removing punctuation, splitting the message into separate words and incrementing counts for each time a word is encountered. Then stop words (e.g. "me", "you", "hers") are removed from the word count list and the N most popular words are chosen for the feature dictionary. Each feature vector is then an array of length N+2 where the first and last values are 0 and 1 and each value in between is the number of times a word in the feature dictionary appears in a given sentence.

The bag of words feature using the N highest TF-IDF words created using the TF (term frequency) and the IDF (inverse document frequency) of each unique word w. The feature dictionary is the same as the previous feature representation: the N most popular words. First, the DF (document frequency) of each word is calculated by counting, for each word, how many messages have at least one occurrence of said word. Then, to construct a message *d*'s feature vector, I calculate the term frequency of each word in the feature dictionary as the number of occurrences of each word in the message *d*. Each term frequency is multiplied by the inverse document frequency of the corresponding word, resulting in the TF-IDF value for the word for the given message. The final feature vector for the message is then an array of length N+2 where the first and last values are 0 and 1 and each value in between corresponds to a word from the feature dictionary and is calculated as

$$tfidf(w,\ d,\ D)\ =\ tf(w,\ d)\ *\ log(\frac{len(D)}{df(w,D)})$$

where *w* is a word, *d* is a message, and *D* is the set of all messages (in this case the training set).

The final feature representation utilizes the word2vec model from the library gensim to create a word embedding vector for each message which is then used as the feature vector for said message. The word2vec model is created by first separating each message into tokenized lists of words. The model is then trained using the tokenized lists by optimizing hyperparameters and comparing the resulting accuracy of a logistic regression model, with set hyperparameters, trained and validated on the training and validation sets after vectorizing them using the word2vec model. The best word2vec model is then used to create word embedding vectors for each message in the training data set which are then used as the final feature vectors.

3 Model Training, Optimization, and Comparison

The models I'll use include sklearn's LogisticRegression model and sklearn's AdaBoostClassifier. Both models are classifiers which fit the predictive task I set for the dataset. Logistic regression is simpler and less expensive to train while AdaBoost is able to leverage the combined predictive power of a large number of weak learners which are each adjusted to improve performance on more difficult cases. This strikes a balance between variety of complexity while also keeping scalability somewhat manageable.

Optimization for each pair of feature representation and model was done by looping over a range of values for each hyperparameter. Each hyperparameter was optimized one at a time rather than looping through all combinations of all hyperparameters in order to reduce the time required to train while still benefiting from some optimization.

The bag of words-based models didn't encounter much issue with overfitting while the word2vec-based LogisticRegression model did run into some overfitting issues, likely due to the word2vec model's training using a LogisticRegression model during validation.

Some other models I considered were sklearn's SGDClassifier which allows training of an SVM model with SGD training, and sklearn's RandomForestClassifier. I ultimately did not successfully train either model due to time constraints. SGDClassifier, which can train an SVM model, has the advantages of optimizing classification error which lets it be non-probabilistic and being more effective in high-dimensional spaces and the disadvantages of being computationally expensive and poor at large datasets [3]. RandomForestClassifier has the advantages of being robust to noise, non-parametric in nature, and leveraging an ensemble approach for higher accuracy while having the disadvantages of being difficult to interpret and computationally and memory expensive [4].

4 Related Literature

There has been lots of active work and research into detecting the presence of hate speech in language as well as considering the importance of mistakes in detection.

The dataset I used for instance, was created by collecting data from various different sources and preprocessing the data for use in hate speech detection models and studies. One example of this is by Behzadidoost et al. in their work on hate speech detection which includes the proposal of a granular computing-based deep learning model and comparison to other state-of-the-art models [5]. Another study by Sharif et al. involves using a larger comprehensive dataset made up of the dataset I used as well as other datasets to train and test a CNN model and a BiLSTM with an attention mechanism [6].

Some other datasets that have been studied include Jigsaw, Contextual Abuse Database (CAD), HateXplain, as well as others [11]. Jigsaw consists of about 230,000 wikipedia comments that were rated for toxic behavior. CAD is an annotated dataset of about 25,000 reddit entries labeled across six conceptually distinct primary categories with rationales for the classification included for each entry. HateXplain is a benchmark hate speech dataset with about 20,000 samples with each sample being given a 3-class classification of (hate, offensive, normal), the target community, and the rationales for the classification.

Other state-of-the-art models used to classify text data as hate speech include BERT, MPNet, ChatGPT, and others. BERT improved off of previous language models by using a masked language model as the pre-training objective which let features represent both the left and right context [7]. MPNet inherits the advantages of BERT and XLNet while trying to avoid their limitations by using permuted language modeling and taking auxiliary position information as additional input [8]. ChatGPT is a large language model based on the GPT-3.5 architecture developed by OpenAI. In their study, Das et al. performed experiments to determine ChatGPT effectiveness at detecting hate speech in numerous languages, especially noting its weaknesses in specific areas [9].

While most state-of-the-art research in recent times utilizes more complex and expensive models, logistic regression models like the ones I trained are still used in some research. For instance, Damayanti et al. propose a combined model that uses a Logistic Regression and a Support Vector Machine to classify hate comments on Twitter [10]. Their results ended up with significantly better performance, namely higher accuracy, precision, recall, and F1 score, showcasing the viability of simpler models like logistic regression.

5 Results and Conclusions

Table 2 showcases the performance of the trained models on the test set.

5.1 Performance

Logistic regression using the N most common words as the feature representation achieved the best accuracy though it was only 0.0004% better than when using the N highest TF-IDF words, so no gain is seen after rounding to 3 significant figures. Logistic regression with word2vec embedding vectors performed the worst on all metrics, though this is likely because of overfitting from the training process of the final word2vec model. The AdaBoost model with word2vec embedding vectors, while having 0.5% lower accuracy and 6.5% lower precision than the models trained using bag of words features, also has a significant 11.5/11.4% higher recall which may be detrimental in real world detection of hate speech as false positives may or may not be considered more important than false negatives.

Bag of words feature representations performed better than the word embedding feature representations which may in part be due to the overfitting issue with the word2vec model. The difference in performance may also be partially due to a difference in hyperparameter optimization as AdaBoost and word2vec both have significantly more hyperparameters that can be optimized compared to using just a LogisticRegression and adjusting the size of the bag of words dictionary.

5.2 Interpretation

The word2vec and the LogisticRegression trained on it and the AdaBoost models' parameters are hard to interpret because they use a large 2D matrix and a large vector of weights, respectively. It is interesting to note that the AdaBoost model's weights include some values of 0, indicating that some of the DecisionTreeClassifiers used for the model's weak learners don't have any contribution to predictions. Of the parameters in the LogisticRegression trained on word2vec embeddings, some values are orders of magnitude higher than the rest, indicating that parts of the embedding are more important than others, though it isn't clear what they represent.

The weights for the LogisticRegression models trained on bag of word features are significantly more interpretable since the

Feature	Model	Accuracy	Precision	Recall	F1	BAR
None	Random	0.5	0.209	0.5	0.295	0.5
BoW N most popular words	Logistic Regression	0.829	0.663	0.365	0.471	0.658
BoW N highest TF-IDF	Logistic Regression	0.829	0.663	0.364	0.470	0.658
word2vec embedding	Logistic Regression	0.817	0.621	0.317	0.420	0.633
word2vec embedding	AdaBoost	0.824	0.598	0.480	0.533	0.698

Table 2: Model performance on test set

parameters correspond to the weight each word in the feature has on the prediction.

For both models that used bag of words representations, most of the highest weighted words were mostly similar and included words like sexist, idiot, suck, as well as other slurs and expletives. The ordering of the highest weighted words between the models using and not using TF-IDF slightly differ, but the same words have similar levels of impact under both representations.

5.3 Conclusions

In this study, I use combinations of different feature representations and models to classify whether a given message is hate speech or not. While all models performed roughly similarly in terms of accuracy, the relatively low values for the other performance metrics indicate that real world use of these models to classify messages would prove difficult due to potentially large numbers of false positives, which may necessitate more manual intervention and checking, and false negatives, which would leave hate speech undetected. Additionally, the models I trained perform worse than more state-of-the-art models used in other studies.

6 References

[1] Mody, Devansh; Huang, YiDong; Alves de Oliveira, Thiago Eustaquio (2022), “A Curated Hate Speech Dataset”, Mendeley Data, V1, doi: 10.17632/9sxpkmm8xn.1

[2] Wendyellé A. Alban Nyantudre (2023), "Hate Speech Detection curated Dataset🤬", https://www.kaggle.com/datasets/waalbannyantudre/hate-speech-detection-curated-dataset/data?select=HateSpeechDataset.csv

[3] goelaparna1520 (2023), Support vector machine in Machine Learning, https://www.geeksforgeeks.org/support-vector-machine-in-machine-learning/

[4] ravinderkamatw (2024), What are the Advantages and Disadvantages of Random Forest?, https://www.geeksforgeeks.org/what-are-the-advantages-and-disadvantages-of-random-forest/

[5] Rashid Behzadidoost, Farnaz Mahan, Habib Izadkhah (2023), Granular computing-based deep learning for text classification, https://doi.org/10.1016/j.ins.2023.119746

[6] Waqas Sharif, Saima Abdullah, Saman Iftikhar, Daniah Al-Madani, Shahzad Mumtaz (2024), Enhancing Hate Speech Detection in the Digital Age: A Novel Model Fusion Approach Leveraging a Comprehensive Dataset, doi: 10.1109/ACCESS.2024.3367281

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, (2019), BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,
https://doi.org/10.48550/arXiv.1810.04805

[8] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, Tie-Yan Liu, (2020), MPNet: Masked and Permuted Pre-training for Language Understanding,
https://doi.org/10.48550/arXiv.2004.09297

[9] Mithun Das, Saurabh Kumar Pandey, Animesh Mukherjee, (2024), Evaluating ChatGPT Against Functionality Tests for Hate Speech Detection,
https://aclanthology.org/2024.lrec-main.564/

[10] Damayanti, N. P., Prameswari, D. E., Puspita, W., & Sundari, P. S. (2024). The Classification of Hate Comments on Twitter Using a Combination of Logistic Regression and Support Vector Machine Algorithm,
https://doi.org/10.52465/joiser.v2i1.229

[11] Christopher Clarke, Matthew Hall, Gaurav Mittal, Ye Yu, Sandra Sajeev, Jason Mars, Mei Chen, (2023), Rule By Example: Harnessing Logical Rules for Explainable Hate Speech Detection,
https://doi.org/10.48550/arXiv.2307.12935